# An OpenCCG-Based Approach to Question Generation from Concepts

Berg, M. M., Isard, A., & Moore, J. D. (2013). An OpenCCG-Based Approach to Question Generation from Concepts. In E. Métais, F. Meziane, M. Saraee, V. Sugumaran, & S. Vadera (Eds.), Natural Language Processing and Information Systems. (pp. 38-52). (Lecture Notes in Computer Science). Springer Berlin / Heidelberg. 10.1007/978-3-642-38824-8_4

# An OpenCCG-based Approach to Question Generation from Concepts

Markus M. Berg[1,2], Amy Isard[3], and Johanna D. Moore[3]

[1] Department of EE & CS, Wismar University, Germany, mail@mmberg.net
[2] Department of Computer Science, Kiel University, Germany
[3] School of Informatics, University of Edinburgh, Scotland, UK, {amy.isard,j.moore}@ed.ac.uk

**Abstract.** Dialogue systems are often regarded as being tedious and inflexible. We believe that one reason is rigid and inadaptable system utterances. A good dialogue system should automatically choose a formulation that reflects the user's expectations. However, current dialogue system development environments only allow the definition of questions with unchangeable formulations. In this paper we present a new approach to the generation of system questions by only defining basic concepts. This is the basis for realising adaptive, user-tailored, and human-like system questions in dialogue systems.

## 1 Introduction

Speech-based information systems offer dialogue based access to information via the phone. They avoid the complexity of computers/websites and introduce the possibility of accessing automated systems without being distracted from your visual focus (e.g. while driving a car) and without needing your hands or eyes (e.g. for visually impaired people or workers with gloves). Most people have already used spoken dialogue systems (SDS) in order to reserve tickets for the cinema, to check the credit of a pay-as-you-go phone or to look for the next bus. Generally, a SDS can be defined as a system that *"enables a human user to access information and services that are available on a computer or over the Internet using spoken language as the medium of interaction"* [11]. In this way, these systems can offer a convenient way of retrieving information. However, Bringert [5] identifies three major problems with current interactive speech applications: They are not natural, not usable and not cheap enough. Berg [2] found that 71% of users prefer the most natural dialogues when choosing from three fictional human-machine dialogues. This is in line with the results of Dautenhahn et al. [7], who also found that 71% of people wish for a human-like communication with robots. Looi and See [14] describe the stereotype of human-robot dialogue as being monotonous and inhumane. They argue that the engagement between human and robot can be improved by implementing politeness maxims, i.e. connecting with humans emotionally through a polite social dialogue. In order to realise user-friendly and natural dialogue systems that apply politeness maxims and adapt their style to the user's language, we need support from a language generation component.

In this paper we describe a method for generating system questions in information-seeking dialogue systems. Our aim is to formulate these questions in different styles (formality and politeness) from abstract descriptions (*concept-to-speech*). We hope to increase the user acceptance of dialogue systems by contributing to a method for generating human-like and adaptive utterances.

## 2 Related Work

As this paper focusses on the generation of questions in different styles, we review related work in the areas of question generation and linguistic style.

### 2.1 Question Generation

Question Generation is a twofold area of research. On the one hand, it deals with text understanding and the generation of questions related to the content. This area is of special interest for tutoring system researchers, where the system has to understand the content of an article, identify relevant sentences and formulate questions about them in order to automatically create reading assessments. On the other hand, question generation can be seen as a subdomain of language generation and *concept-to-speech* technology. Especially in rapid application development, the specification of concepts along with surface parameters instead of hard-coded system prompts reduces development effort and introduces the possibility of adapting system utterances with regard to the demands of the user. Based on the contributions to the *Workshops on Question Generation* in 2009 [21] and 2010 [4], most work has been done in the area of tutorial dialogue, i.e. question generation from text. Papasalouros [19] describes how to generate multiple choice questions for online examination systems, Ou et al. [18] show how to extract predictive questions from an ontology that might be asked by the user of a question-answering system, and Olney et al. [17] describe an approach to generate questions by filling templates. Their approach also focusses on tutorial dialogue and is based on psychological theories that claim that questions are generated from a concept map. But as the use of representations has been avoided in state of the art question answering systems, Olney et al. discuss whether a representation-free approach that bases on syntax transformations (e.g. wh-fronting) of given declarative sentences can successfully generate questions. They found that this approach has difficulties with determining the question type and conclude that a certain degree of knowledge representation is necessary.

### 2.2 Linguistic Style

Mairesse [15] explains linguistic style as *"a specific point within the space of all possible linguistic variation"* and concludes that it can be considered as a *"temporary characteristic of a single speaker"*. He refers to Brown et al. [6] and the need for self-esteem and respect from others and states *"that the use of politeness is dependent on the social distance and the difference of power between conversational partners, as well as on the threat of the speakers communicative act towards the hearer"*. Gupta et al. [8] state that *"Politeness is an integral part of human language variation, e.g. consider the difference in the pragmatic effect of realizing the same communicative goal with either 'Get me a glass of water mate!' or 'I wonder if I could possible have some water please?' "*. Jong et al. [12] claim that language alignment happens not only at the syntactic level, but also at the level of linguistic style. They consider linguistic style variations as an important factor to give virtual agents a personality and make them appear more socially intelligent. Also Raskutti and Zukerman [20] found that naturally occurring information-seeking dialogues include social segments like greeting and closing. Consequently, Jong et al. [12] describe an alignment model that adapts to the user's level of politeness and formality. The model has three dimensions: politeness, formality and T-V distinction. Whereas politeness is associated with sentence structures, formality is dependent on the choice of words. In many languages we have to differentiate between a formal and an informal addressing (T-V distinction) of people. This feature is clearly related to both formulation and politeness. However, in Jong's model this feature is not being influenced by formality or politeness changes during the conversation in order to prevent the dialogue from constantly switching between both extrema.

## 3 Style Variation

Style variation is the generation of different formulations with the same goal. In this paper we focus on task-oriented dialogue systems. This class comprises question-answering-, command-,

and information-seeking/booking systems [16]. In particular, we regard the style variation of *interrogatives*. We use this term in order to refer to all kinds of utterances that have the aim of getting information. This may be a question (*"When do you want to leave for London?"*) or a request (*"Tell me when you want to leave for London!"*). Both interrogatives have the same intention, i.e. getting the time of departure.

### 3.1 Question Types

When trying to classify questions, we have to differentiate between intention and style. When classifying by intention, both interrogatives from the last example should be of the same type. Berg et al. [3] describe an Abstract Question Description (AQD) that consists of the answer type, reference type, purpose, surface modifier and cardinality. This would result in `AQD = (fact.temporal.date, fact.namedEntity.nonAnimated.location.city, gather information, positive, 1)`, i.e. an interrogative that expects a date as answer, refers to a city, is meant to get new information from the user, refers to a positive formulation, and expects a single answer. This very abstract definition is useful when it comes to the modelling of dialogue systems. Imagine an integrated development environment that allows you to define concepts from which the system should generate questions, instead of formulating static and inflexible questions as strings. In this scenario, AQDs can also formalise the parser (in this case a date grammar could be provided). They also help the language understanding component by reducing differently formulated utterances with the same goal to a common description.

For question generation, however, we need more information about the style. While the classification by question words has been declared impractical for describing the intention of an interrogative because different question words can refer to the same goal, e.g. *when* and *at what time* [3], it is still important for style variation. Hence, the relation between AQD and question word can be useful for choosing the correct question word. Apart from the question word, we can also vary grammatical characteristics. When generating a question for the AQD answer type `fact.temporal.date`, we can think of different formulations:

**Wh-Question:** *When do you want to go?*
**Wh-Request:** *[Please] Tell me / specify when you want to go!*
**NP-Request:** *[Please] Tell me your departure time [please]!*
**C-Wh-Question:** *Can/Could you [please] tell me when you want to go?*
**C-NP-Question:** *Can/Could you [please] tell me your departure time?*
**Command (NP):** *Your departure time?*
**Command (N):** *Departure time...?*

We clearly see that all these utterances have the same intention. However, the AQD is not sufficient for successful generation. In addition to the AQD we also need to define semantic constraints. In this case we could constrain the type of date to departures.

### 3.2 Politeness and Formality

We have seen different realisations of the same message. But how do these formulations affect us and how is this related to politeness and formality? We have already learned that there exist different degrees of politeness and that systems with an appropriate choice of style are important to the user. However, it is hard to tell what exactly makes an utterance more polite or more formal than another. Moreover, formality and politeness are very close terms that often get mixed. Often a question (*"When do you want to go?"*) seems more polite than a request (*"Tell me when you want to go!"*). But what if we add *please* to the request? Is the request more polite

because of this modal particle? Is *can you please* more polite than *could you*? In this paper we work with the following hypothesis, which we plan to evaluate in future user studies: Politeness is characterised by:

- the use of *please*
- the use of a subjunctive modal verb
- an interrogative style (request vs. question)

Formality is sometimes also regarded as influencing politeness, i.e. a very formal style is intended to be polite. In most cases this is true, but nevertheless we have to distinguish politeness and formality in order to be able to adapt to it independently. We regard formality as the lexicalisation, i.e. the choice of words and word types.

*Can you tell me when you like to set off?*
*Can you tell me when you want to leave?*
*Can you tell me your departure date?*
*Can you specify when you want to leave?*
*Can you specify your departure date?*

In this example above we can identify different levels of formality in relation to the choice of words, i.e. *specify* sounds more formal than *tell* and *set off* sounds more colloquial than *leave*. Moreover, the grammatical structure can also influence formality. While the first two sentences use a verb phrase (*you want to leave*), the third one makes use of a noun phrase (*your departure date*). Some languages (e.g. German or French) also differentiate between formal and informal personal pronouns (second person). This is called T-V distinction and affects, as an indicator for social distance, formality. Again, the exact ordering of utterances regarding formality is subject for a user study. We work with the hypothesis that formality can be influenced by:

- the choice of words
- the grammatical form of the sentence (NP vs VP)
- the choice between formal and informal personal second person pronouns

With this distinction between politeness (the use of *please* and subjunctive forms, choice of question style) and formality (choice of words, T-V distinction) we now have parameters at hand to change the style of a system interrogative. In the next section we address the topic of modelling system interrogatives for usage in a toolkit that allows the realisation of interrogatives with respect to given parameters.

## 4 Realisation

Our aim is the automatic generation of system questions in spoken dialogue systems. We want to provide the development environment with information like *"ask for the departure date in an informal way and don't make special use of politeness"* and it should generate a question like *"When do you want to leave?"*. This process includes several steps and components. As you can see in Figure 1, we first need to define the communicative goal. This goal needs to be connected to the content (what should be said) and the form (how it should be formulated). This is known as lexicalisation (choice of words and grammatical style) and dependent on the parameters for politeness and formality. This decision has to be made in close connection to the content, i.e. we need to find a representation that provides us with information about word meaning, formality and politeness. Moreover, we need a set of rules for the generation of different question types. With this information we can create logical forms that can be used as input for our language generator. We first take a closer look at the language generation process. Afterwards we describe our knowledge base and how we can use the results in dialogue systems.

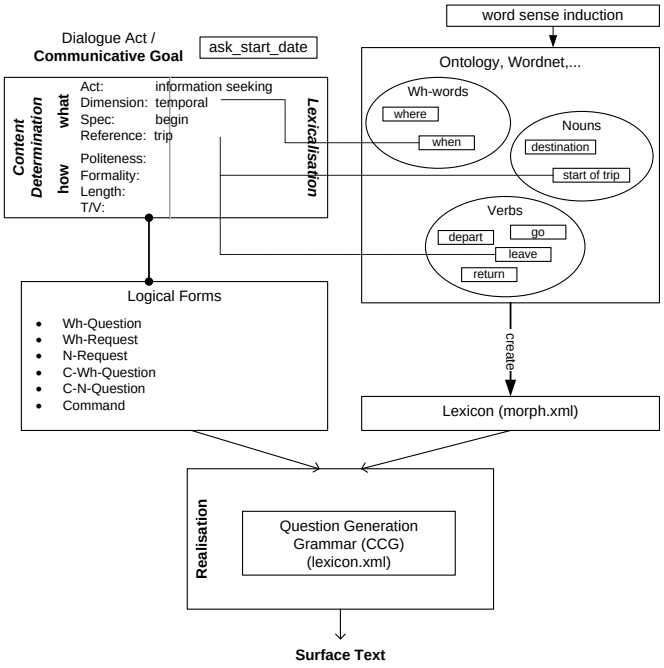



**Fig. 1.** System overview

### 4.1 Combinatory Categorial Grammar

Mark Steedman's Combinatory Categorial Grammar (CCG) formalism contains no phrase structure rules and consists only of lexical entries. Steedman and Baldridge [22] describe this as *"a form of lexicalized grammar in which the application of syntactic rules is entirely conditioned on the syntactic type, or category, of their inputs"*. These categories *"may be either atomic elements or (curried) functions which specify the canonical linear direction in which they seek their arguments"* [1]. Atomic elements (sometimes called saturated), such as N, NP, PP or S, do not need to be combined with other elements. They are complete in themselves. With the help of the slash operator we can combine saturated and unsaturated elements to new saturated or unsaturated elements. The slash indicates on which side the arguments should appear. This leads to forward (argument on the right) and backward (argument on the left) application. We call unsaturated elements functions because a complex category like Y/X denotes an element that looks for an X on the right in order to become an Y. This is exactly the behaviour of a function that takes X as an argument and returns Y as result.

$$\frac{Y/X \quad X}{Y}> \qquad\qquad \frac{X \quad Y\backslash X}{Y}< \tag{1}$$

Apart from application combinators, there are composition and type-raising combinators. Composition refers to the combination of two functions where the domain of one is the range of the second. It is described with the operator $B$ together with the application direction. Type-raising turns *"arguments into functions over functions-over-such-arguments"* [22], i.e. argument $X$ is turned into a function that has a complex argument that takes this $X$ as an argument. It allows *"arguments to compose with the verbs that seek them"* and is also used in order to be

able to apply all rules into one direction (i.e. incremental processing; a full left-to-right proof). Type-raising is denoted with a $T$. Given the following lexicon, we can derive the sentence *"I like science"* as in (2), or with type-raising and composition as a full left-to-right-proof as in (3).

$$\begin{array}{rcl} \textit{science} & \vdash & NP \\ I & \vdash & NP \\ \textit{like} & \vdash & (S\backslash NP)/NP \end{array}$$

(2)

$$\dfrac{\dfrac{i}{\mathsf{np}} \quad \dfrac{\dfrac{\textit{like}}{\mathsf{s\backslash np/np}} \quad \dfrac{\textit{science}}{\mathsf{np}}}{\mathsf{s\backslash np}}>}{\mathsf{s}}<$$

(3)

$$\dfrac{\dfrac{\dfrac{\dfrac{i}{\mathsf{np}} \quad \dfrac{\dfrac{\textit{like}}{\mathsf{s\backslash np/np}} \quad \dfrac{\textit{science}}{\mathsf{np}}}{\mathsf{s\backslash np}}>}{\mathsf{s/s\backslash np}}>\mathbf{T}}{\mathsf{s/np}}>\mathbf{B}}{\mathsf{s}}>$$

OpenCCG[4] is a collection of natural language processing tools, which provide parsing and realisation support based on the CCG formalism. A set of XML-based files allows us to define the lexicon and the categories. For a deeper introduction to the OpenCCG syntax you may refer to [13].

### 4.2 Definition in OpenCCG

In order to generate different formulations, we first have to define a grammar. In this paper we restrict ourselves to the following interrogative types: Yes-No-Question, Wh-Question, Wh-Request, NP-Request, Can-Wh-Question, Can-NP-Question and Command (NP). As already mentioned in the previous section, a categorial grammar does not consist of phrase structure rules that define how a wh-question or a wh-request can be created. Instead, we define a lexicon that describes how every word type can be combined, i.e. how categories can be aggregated to a more general category. A wh-word can be used to create an ordinary question ( *"When do you want to go?"*) or can also be part of an indirect request ( *"Can you tell me when you want to go?"*):

```
when ⊢
    s[wh-question]/s[question] ▷ question
    s[iwh-question]/s[b] ▷ indirect request
```

In our example we apply the first category[5]. This can be read as: *The word 'when' can become a sentence of type 'wh-question' if there is a sentence of type 'question' on the right-hand side.* Since the definition of wh-words is not enough, we now take a look at how the right-hand category of the rule is defined. A question can be created with the help of an auxiliary verb like *do*:

```
do ⊢
    s[question]/s[b]
```

The word 'do' can become a sentence of type 'question' if there is a sentence with a bare infinitive on the right. In order to create such a sentence, we need a verb. Generally, an intransitive verb is defined as `s\np`. The feature `b` denotes a *bare infinitive* and `to` is an *infinitive with to* [10, 9]. The combination of the lexems *want*, *to*, and *go* leads to a category that becomes a sentence with a bare infinitive if there is a `np` on the left. Figure 2 shows the complete application of the CCG categories.

[4] `http://sourceforge.net/projects/openccg/`

[5] brackets indicate features that need to be unified

```
(lex)  when :- s[wh-question]/s[question]
(lex)  do :- s[question]/s[b]
(lex)  you :- np
(lex)  want :- (s[b]\np)/(s[to]\np)
(lex)  to :- s[to]\np/(s[b]\np)
(lex)  go :- s[b]\np
(>)    to go :- s[to]\np
(>)    want to go :- s[b]\np
(<)    you want to go :- s[b]
(>)    do you want to go :- s[question]
(>)    when do you want to go :- s[wh-question]
```

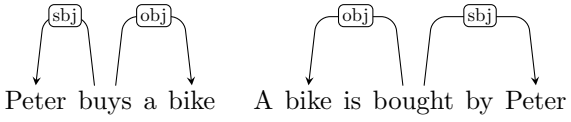
**Fig. 2.** Simplified parse for 'When do you want to go?'

In this example we have described the general methodology to generate wh-questions. However, we don't want to parse sentences with the grammar but instead want to generate (or realise) them. The basis for the OpenCCG realisation process is logical forms, i.e. semantic representations of the sentence. Each syntactic category is associated with a logical form represented in a *hybrid logic dependency structure* that describes the relations between the words of a sentence; e.g. the transitive verb *buy* can be described as:

$$@e\ buy,\ @e{<}sbj{>}\ x,\ @e{<}obj{>}\ y$$

The verb *buy* has two relations, subject and object. The advantage of this semantic description is its independence from the syntactic form. The sentences *"Peter buys a bike"* and *"A bike is bought by Peter"* can be represented as:

$$@e(buy \wedge \\ <sbj>\ (x \wedge Peter) \wedge \\ <obj>\ (y \wedge bike))$$




The logical form for the wh-question from our last example is depicted in Figure 3. We use thematic roles in order to specify the proposition of a question. In this case an *agent* wants a *theme*. We can also use semantic features to influence the style of the utterance, i.e. in this case we want to formulate an interrogative sentence with a second person singular agent. As already

```
s{stype=wh-question}:
  @w0(when ^
      <prop>(w3 ^ want ^
             <mood>interrogative ^
             <agent>(w2 ^ pron ^
                    <num>sg ^
                    <pers>2nd) ^
             <theme>(w5 ^ go ^
                    <agent>x1)))
```

**Fig. 3.** Logical form for 'When do you want to go?'

mentioned, logical forms are the basis for the realisation process. It abstracts from grammar and word position issues and just reflects the logical meaning of an utterance. Additionally, we need to know which words to use, i.e. OpenCCG requires a finished lexicalisation process. While there are words that are only influenced by inflection (pronoun, sg, 2nd), we also have words that change the style of an utterance[6]. Another way of realising the same meaning with a different lexicalisation would have been *"When do you want to leave?"*.

## 5 Concept to Text

As already mentioned, our aim is the automatic generation of system questions in an information-seeking dialogue system. We want to be able to instruct the system to create *a question that asks for the departure time in a polite but informal way* without mentioning specific words. This is absolutely necessary to create different levels of formality. So instead of defining words in the logical form we need meaning representations, i.e. we have to replace the bold faced words in Figure 3 with concepts.

### 5.1 Word Meaning

In order to describe question words, we can use a slightly modified version of Berg's AQD hierarchy [3], e.g. *"when"* can be described by `fact.temporal`, *"where"* by `fact.named_entity.non-animated.location` and *"at what time"* by `fact.temporal.time`. For verbs we need more information than just the dimension. One approach is the description of the meaning of a word by combining conjunctive features:

- go: movement $\wedge$ slow $\wedge$ by feet
- run: movement $\wedge$ fast $\wedge$ by feet
- drive: movement $\wedge$ by car
- travel: movement $\wedge$ far away $\wedge$ holiday

However, we need a more abstract formulation that also focusses on the similarities of word senses. In a travel domain *go* and *travel* can be synonymous (*"When do you want to go"* = *"When do you want to travel"*) and should therefore have the same description. Thus, as a first draft, we propose to describe a word with its type of usage (or context), so that every word $w$ is assigned a:

- Part of Speech $\pi$
- Domain $\delta$
- Context $\gamma$: Dimension $\wedge$ Specification
- Referent $\rho$

which results in ($w$ typeof $\pi$) $\in \delta = (\gamma, \rho)$. In the following examples you can see how we can describe the meaning of words, together with some exemplary sentences. The definition of *go* reads as follows: It is a verb in the travel domain that can be used in a temporal context to describe the beginning of a trip (start date) or in a local context to describe the end of a trip (destination). Nouns follow the same scheme. The only difference is the part of speech.

[6] indicated with bold face in Figure 3

**go** (v) ∈ Travel
$\gamma$=temporal∧begin ∨ local∧end
$\rho$=trip
▷ *When do you want to go?*
▷ *Where do you want to go?*

**departure city** (n) ∈ Travel
$\gamma$=local∧begin
$\rho$=trip
▷ *Please tell me your departure city.*

When we take a look at question words, we can see that we have introduced a *general* domain and a wildcard referent as well as a wildcard specificator.

**when** (whadv) ∈ General
$\gamma$=temporal∧*
$\rho$=*
▷ *When do you want to go?*

**tell** (v) ∈ General
$\gamma$=*∧knowledge_transfer
$\rho$=*
▷ *Can you tell me when you want to go?*

Apart from question words and related verbs or nouns (*when* do you want to *go*) we also have verbs like *want, tell, have* and *can* that cannot be related to an answer type. Here we can use the specificator to denote the context, e.g. a word that can be used in any dimension to indicate a *transfer of knowledge* with reference to any object. With this elementary definition of words we are now able to describe questions. We basically describe a question by $\gamma$ and $\rho$, i.e. the context and the referent. The task of creating a question that asks for the begin of a trip would be defined as: $ask(fact.temporal.date, begin, trip)$. According to our question style definitions from section 4.2 and their related grammars, we choose either a verb or a noun to represent $\gamma$ and $\rho$. Also neutral words like *want* or *tell* are introduced in this step. In a very formal utterance *tell* could be replaced by *specify*. Apart from the formality, we also have to choose the correct question style according to the politeness. A high politeness value leads to the introduction of the word *please* and changes the mood from indicative to subjunctive. Moreover, we have a list that assigns a politeness value to every question type and thus influences the construction of the logical forms. For example, a can-question is more polite than a request. These values are currently based on intuitions which were backed up by the choices of the evaluation participants (see Section 6) but in future versions we plan to base them on user studies.

The result of this step is the representation in Figure 3, which is – at the same time – the input for the OpenCCG realiser.

### 5.2 Programming Interface & Results

As mentioned in the beginning, our aim is an easy integration of our approach in current dialogue systems. The programmer should not have to deal with complicated language generation issues. Instead, he should be able to use a simple interface. The management of the vocabulary is handled by an ontology. The following lines of code generate a dialogue with five system questions, that ask for the start date of the trip, the end date, the departure city, the destination and whether the customer has a customer card:

```
1 int intended_formality=2;  //1..5
2 int intended_politeness=1; //-2..5
3 questions.add(new Question("fact.temporal.date", "begin", "trip"));
4 questions.add(new Question("fact.temporal.date", "end", "trip"));
5 questions.add(new Question("fact.location", "begin", "trip"));
```

```
6 questions.add(new Question("fact.location", "end", "trip"));
7 questions.add(new Question("decision", "possession", "customer_card"));
```

When setting the formality value to 2 and the politeness value to 1, we achieve the result shown in Dialogue 1. As you can see, every second utterance we insert a temporal connector (*now*) to make the dialogue appear more fluent. In Dialogue 2 we have increased the politeness value to 4. You can see that the system now chooses C-Questions and makes use of verbs instead of nouns.

Dialogue 1: *f=2, p=1*

*S: Please tell me your departure date!*
*U: ...*
*S: Now please tell me your return date!*
*U: ...*
*S: Please tell me your departure city!*
*U: ...*
*S: Now please tell me your destination!*
*U: ...*
*S: Do you have a customer card?*
*U: ...*

Dialogue 2: *f=2, p=4*

*S: Can you please tell me when you want to go?*
*S: Can you now please tell me when you want to return?*
*S: Can you please tell me where you want to start from?*
*S: Can you now please tell me where you want to go?*
*S: Do you have a customer card?*

When also increasing the formality to 4, we observe a different choice of words, for example the first utterance may be realised as *"Can you please tell me when you want to depart?"*. We can see that due to the static relationship between formality and question style, almost every utterance has the same formulation within each dialogue. That's why we introduce a *politeness* variation that automatically varies the politeness around a given value, as you can see in Dialogue 3.

Dialogue 3: *f=2, p=1±1*

*S: Please tell me your departure date!*
*S: And when do you want to return?*
*S: Departure city please!*
*S: Now please tell me your destination!*
*S: Do you have a customer card?*

## 6 Evaluation of Generated Example Interrogatives

After having demonstrated the functionality of the proposed system, we now evaluate its plausibility and effects. We have asked 26 human judges to evaluate the dialogues' naturalness and politeness. During the test we presented the participants four dialogues (see Figure 4) with different politeness levels. The users had to sort the dialogues according to their politeness. Afterwards, they had to indicate which of the dialogues might have been uttered by a human and to state which dialogue they prefer.

We began with the sorting task. In general the participants correctly classified the dialogues according to our intended politeness levels, and 46% put the dialogues in exactly the right order $(C, D, A, B)$. The participants classified the two more impolite dialogues as more polite than they are, and the two more polite ones as less polite, as shown in Table 1, but this can possibly be explained by people's tendency to choose values in the middle of a scale rather than at either extreme. Now we asked the participants which dialogue they like most. 77% of them preferred dialogue A, 19% preferred dialogue B and 4% dialogue D. When we quantify the preferred dialogues with the corresponding politeness scores and normalise the result (see Equation 4),

| **A** | **B** | **C** | **D** |
|---|---|---|---|
| A: When do you want to set off?<br>B: ...<br>A: Can you now tell me when you want to return?<br>B: ...<br>A: Please tell me your departure city!<br>B: ...<br>A: And where do you want to go?<br>B: ...<br>A: Do you have a customer card?<br>B: ... | A: Can you please tell me when you want to set off?<br>B: ...<br>A: Could you now please tell me when you want to return?<br>B: ...<br>A: Can you tell me where you want to start from?<br>B: ...<br>A: Can you now please tell me where you want to go?<br>B: ...<br>A: Do you have a customer card?<br>B: ... | A: Departure date?<br>B: ...<br>A: And the return date?<br>B: ...<br>A: Departure city?<br>B: ...<br>A: And the destination?<br>B: ...<br>A: Customer card?<br>B: ... | A: Departure date please!<br>B: ...<br>A: Now please tell me your return date!<br>B: ...<br>A: Tell me your departure city!<br>B: ...<br>A: And the destination please!<br>B: ...<br>A: Do you have a customer card?<br>B: ... |

**Fig. 4.** Survey

**Table 1.** Dialogues sorted by politeness scores

| *dialogue* | *original scores* | *mean user scores* | $\Delta$ | *correctly classified by* |
|---|---|---|---|---|
| C | 1.0 | 1.8 | +0.8 | 62% |
| D | 2.0 | 2.2 | +0.2 | 62% |
| A | 3.0 | 2.7 | -0.3 | 58% |
| B | 4.0 | 3.3 | -0.7 | 65% |

we get an average preferred politeness score of 3.2 (original score) respectively 2.8 (user score), which again refers to dialogue A.

$$\frac{1}{|user|} \sum_{i=1}^{|dialogues|} score_i \times |votes\_dialogue_i| \tag{4}$$

In the last step we asked the users to indicate which dialogue might have been uttered by a human. 88% of the participants think that dialogue A could have been uttered by a human, 42% think dialogue B might be of human origin, 19% declare dialogue C and 4% dialogue D as human. This evaluation confirms that the system is able to create questions in different politeness levels and that these levels are correctly identified by the users.

## 7 Conclusion and Future Work

We have proposed a model to create system utterances in different politeness and formality levels, which serves as our basis for improving dialogue systems and dialogue development environments with respect to adaptive and human-like formulations of system utterances. Based on a categorial grammar we have created seven different interrogative utterance types that represent the syntactic form for different politeness values. An ontology defines the meanings and formality levels of the used words. To verify the operationality of our model, we have developed a programming interface that is used to define concepts which are then realised according to the politeness and formality parameters. A final evaluation of the generated utterances proves that our model is a valid methodology to support concept-to-text with respect to the generation of system questions.

In our future work we want to extend this model to more complicated utterances and also to support the semi-automatic parser generation from the existing concepts and corresponding answer types. Another area of research is, apart from the realisation of questions, the generation of system answers.

## References


1. Baldridge, J., Kruijff, G.J.M.: Multi-modal combinatory categorial grammar. pp. 211–218. Proceedings of the tenth conference on EACL, Stroudsburg, PA, USA (2003)
2. Berg, M.: Survey on Spoken Dialogue Systems: User Expectations Regarding Style and Usability. XIV International PhD Workshop, Wisla (Poland) (Oct 2012)
3. Berg, M., Düsterhöft, A., Thalheim, B.: Lecture Notes in Computer Science, vol. 7337, chap. Towards Interrogative Types in Task-Oriented Dialogue Systems, pp. 302–307. Springer Berlin / Heidelberg (2012)
4. Boyer, K.E., Piwek, P. (eds.): Proceedings of QG2010: The Third Workshop on Question Generation. Pittsburgh (2010)
5. Bringert, B.: Programming language techniques for natural language applications (2008)
6. Brown, P., Levinson, S.C., Gumperz, J.J.: Politeness: Some Universals in Language Usage. Studies in Interactional Sociolinguistics, Cambridge University Press (1987)
7. Dautenhahn, K., Woods, S., Kaouri, C., Walters, M.L., Koay, K.L., Werry, I.: What is a robot companion - friend, assistant or butler? pp. 1488–1493 (2005)
8. Gupta, S., Walker, M.A., Romano, D.M.: Generating politeness in task based interaction: An evaluation of the effect of linguistic form and culture. pp. 57–64. Proceedings of the Eleventh European Workshop on NLG, Stroudsburg, PA, USA (2007)
9. Hockenmaier, J., Steedman, M.: CCGbank: User's Manual. Tech. rep. (May 2005)
10. Hockenmaier, J., Steedman, M.: CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. Comput. Linguist. 33(3), 355–396 (Sep 2007)
11. Jokinen, K., McTear, M.F.: Spoken Dialogue Systems. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers (2009)
12. Jong, M.d., Theune, M., Hofs, D.: Politeness and alignment in dialogues with a virtual guide. pp. 207–214. Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, Richland, SC (2008)
13. Kruijff, G.j.M., White, M.: Specifying Grammars for OpenCCG: A Rough Guide (2005)
14. Looi, Q.E., See, S.L.: Applying politeness maxims in social robotics polite dialogue. pp. 189–190. Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, ACM, New York, NY, USA (2012)
15. Mairesse, F.: Learning to Adapt in Dialogue Systems: Data-driven Models for Personality Recognition and Generation. Ph.D. thesis, University of Sheffield (Feb 2008)
16. McTear, M.F., Raman, T.V.: Spoken Dialogue Technology: Towards the Conversational User Interface. Springer (2004)
17. Olney, A.M., Graesser, A.C., Person, N.K.: Question generation from concept maps. Dialogue and Discourse 3(2), 75–99 (2012)
18. Ou, S., Orasan, C., Mekhaldi, D., Hasler, L.: Automatic Question Pattern Generation for Ontology-based Question Answering. In: The Florida AI Research Society Conference. pp. 183–188 (2008)
19. Papasalouros, A.: Automatic generation of multiple-choice questions from domain ontologies. Engineer (Bateman 1997) (2008)
20. Raskutti, B., Zukerman, I.: Generating queries and replies during information-seeking interactions. Int. J. Hum.-Comput. Stud. 47(6), 689–734 (Dec 1997)
21. Rus, V., Lester, J. (eds.): AIED 2009 Workshops Proceedings: The 2nd Workshop on Question Generation. Brighton (2009)
22. Steedman, M., Baldridge, J.: Combinatory Categorial Grammar, chap. Non-Transformational Syntax, pp. 181–224. Wiley-Blackwell (2011)